Plan Integral de Pruebas (Funcionalidad, Rendimiento, Alta Disponibilidad y Seguridad)

Propósito: Este plan define el alcance, estrategia, casos, secuencia de ejecución, comandos y criterios de aceptación para validar la plataforma a nivel funcional, rendimiento, resiliencia/alta disponibilidad y seguridad, incluyendo pentesting, escaneo de vulnerabilidades y verificación de hardening.

1. Alcance

- Frontend CMS (WordPress/Drupal) detrás de HAProxy con VIP administrado por Keepalived.
- Servicios en Docker Swarm (web/app, workers, colas si aplican).
- Bases de datos MariaDB (replicación asíncrona / Galera síncrona).
- Almacenamiento compartido con GlusterFS.
- Caché (Varnish/Memcached) y CDN (Cloudflare).
- Observabilidad (logs, métricas, alertas).

2. Reglas de Enganche (Rules of Engagement)

- Pruebas disruptivas (stress, failover, caos, scans intrusivos) se ejecutan en STAGING con paridad de configuración.
- Autorización formal para escaneos y pentesting. Ventanas de mantenimiento acordadas.
- No ejecutar DoS sobre producción. Limitar tasas y duración.
- Canal de comunicación para reportar hallazgos críticos inmediatamente.

3. Criterios de Aceptación Generales

- Disponibilidad ≥ 99.9% en pruebas de resiliencia; sin pérdida de datos.
- Tiempo de respuesta P95 dentro de SLO bajo carga nominal definida.
- 0 vulnerabilidades críticas/altas abiertas al cierre.
- Configuración de seguridad cumple recomendaciones críticas de CIS/Docker/OWASP.

4. Entornos, Datos y Accesos

- STAGING con dominios *.stg.tu-dominio.com y certificados válidos.
- Datos sintéticos/anonimizados; usuarios de prueba (admin/editor/autor/lector).

5. Matriz de Pruebas (Resumen)

ID	Área	Objetivo	Herramientas	Criterio de Éxito
FUN-01	Funcional	CRUD de contenido, login, media	Selenium,cURL	Flujos OK, sin errores
PERF-	Rendimient	Baseline/Spike/Stress/Soa	JMeter	P95 dentro de

01	o	k		SLO
HA-01	Alta Dispon.	Failover sin downtime perceptible	Docker,Keepalived	RTO < 30s
SEC-01	Pentesting	DAST, SQLi, XSS, headers/TLS	ZAP,Nikto,sqlmap	0 críticas/altas
VULN-01	Vuln Mgmt	CVEs en hosts/containers	OpenVAS/Nessus	0 críticas abiertas
CONF-01	Hardening	Host y contenedores seguros	Lynis,Docker Bench	Cumplimiento CIS crítico
DB-01	Base de Datos	Consistencia, failover, permisos	MariaDB CLI	Sin pérdida de datos
FS-01	Storage	Consistencia GlusterFS, latencia	gluster,fio	Lectura/escritura OK
CDN-01	CDN/Cache	Hits, invalidación, headers	curl,Cloudflare	Hit ratio y seguridad OK
CHAOS-01	Caos	Degradación controlada de red/CPU	tc,stress-ng	SLOs sin caída

6. Casos y Procedimientos de Prueba con Comandos

6.1 Pruebas Funcionales (CMS)

Validar rutas críticas, autenticación y edición concurrente en múltiples réplicas.

Comandos útiles (HTTP y headers):

```
# Verificar 200 OK, redirecciones y headers de seguridad
curl -I https://staging.tu-dominio.com/
curl -s -o /dev/null -w "%{http_code}
" https://staging.tu-dominio.com/login
# Probar subida de archivo (media)
curl -F "file=@/path/a/logo.png" -b cookies.txt -c cookies.txt
https://staging.tu-dominio.com/wp-admin/async-upload.php
```

6.2 Rendimiento (Apache JMeter)

Escenarios: Baseline, Spike, Stress, Soak. Ejecutar en modo no-GUI.

Comandos JMeter:

```
# Baseline (propiedad BASE_URL)
jmeter -n -t security_load_scenarios.jmx -JBASE_URL=https://staging.tu-
```

```
dominio.com -l results.jtl -e -o ./report
# Spike: ajustar en el .jmx o usar -Jusers/-Jramp (si parametrizado)
# Merge de resultados de múltiples nodos (opcional) con JMeter-Plugins CMDTool
```

6.3 Alta Disponibilidad y Failover (HAProxy, Keepalived, Swarm)

Procedimientos y comandos:

```
# Estado de HAProxy (estadísticas vía socket/admin si habilitado)
echo "show info" | socat stdio /var/run/haproxy/admin.sock
# Verificar VIP y prioridad de Keepalived
ip addr | grep -A2 'vrrp'
# Forzar failover (bajar prioridad en MASTER temporalmente)
sudo systemctl stop keepalived

# Docker Swarm - estado y reprogramación
docker node ls
docker service ls
docker service ps web_app
# Simular caída de nodo worker (cordon + drain)
docker node update --availability drain worker-02
# Escalar servicio web
docker service scale web_app=6
# Actualizar imagen con despliegue rolling y límites de recursos
docker service update --image repo/web:stable --limit-cpu 1 --limit-memory
512M --update-parallelism 1 --update-delay 30s web_app
```

6.4 Límites de Recursos en Contenedores (CPU/RAM) y Recovery

Configurar límites y observar comportamiento ante saturación:

```
# Crear/actualizar servicio con límites
docker service create --name api --limit-cpu 0.5 --limit-memory 256M
repo/api:latest
docker service update --limit-cpu 0.5 --limit-memory 256M api
# Monitoreo rápido
docker stats --no-stream
# Inyectar carga de CPU/RAM en un contenedor (usar en STAGING)
docker exec -it $(docker ps -q -f name=api) sh -c "apk add --no-cache stress-
ng && stress-ng --cpu 2 --timeout 120s"
```

6.5 Base de Datos (MariaDB InnoDB/Galera)

Verificación de replicación, permisos y failover:

```
# Conexión y estado
mysql -h db-vip -u monitor -p -e "SHOW VARIABLES LIKE 'version';"
# Replicación asíncrona
mysql -e "SHOW SLAVE STATUS\G" | egrep
'Slave_IO_Running|Slave_SQL_Running|Seconds_Behind_Master'
```

```
# Galera (síncrona)
mysql -e "SHOW STATUS LIKE 'wsrep_cluster_size';"
mysql -e "SHOW STATUS LIKE 'wsrep_local_state_comment';"
# Backups y restauración (ejemplo con mariabackup)
mariabackup --backup --target-dir=/backup/$(date +%F)
mariabackup --prepare --target-dir=/backup/2025-08-19
mariabackup --copy-back --target-dir=/backup/2025-08-19
```

6.6 Almacenamiento Compartido (GlusterFS)

Consistencia, rendimiento y resiliencia:

```
# Comprobar estado del clúster y volúmenes
gluster peer status
gluster volume status
gluster volume info
# Self-heal y split-brain
gluster volume heal VOL_NAME info
# Prueba de I/O (fio) - usar con cuidado
fio --name=randrw --rw=randrw --bs=4k --size=1G --iodepth=16 --numjobs=4 --
runtime=120 --group_reporting --filename=/mnt/gluster/testfile
```

6.7 CDN (Cloudflare) y Caché (Varnish/Memcached)

Validar headers, purga e hit ratio:

```
# Verificar headers de caché (Cloudflare/Varnish)
curl -I https://staging.tu-dominio.com/ | egrep 'cf-cache-status|cache-
control|age|x-cache|via'
# Invalidar objeto en CDN (ejemplo con API token exportado en CF_TOKEN)
curl -X POST "https://api.cloudflare.com/client/v4/zones/$CF_ZONE/purge_cache"
-H "Authorization: Bearer $CF_TOKEN" -H "Content-Type: application/json"   --
data '{"files":["https://staging.tu-dominio.com/path/asset.css"]}'
# Prueba contra Varnish admin (si expuesto a red de gestión)
varnishstat -1 | egrep 'MAIN.cache_hit|MAIN.cache_miss'
```

6.8 Seguridad – Pentesting Interno (OWASP ZAP, Nikto, sqlmap)

Ejecución controlada en STAGING. Incluir usuarios de prueba y rutas autenticadas.

```
# ZAP Baseline (Docker)
docker run --rm -t -v $PWD:/zap/wrk owasp/zap2docker-stable   zap-baseline.py
-t https://staging.tu-dominio.com -r zap_baseline.html --auto

# ZAP Full Scan (intrusivo, usar con contexto/autenticación)
docker run --rm -t -v $PWD:/zap/wrk owasp/zap2docker-stable   zap-full-scan.py
-t https://staging.tu-dominio.com -r zap_full.html -z "-config
scanner.threadPerHost=5 -addonupdate"

# Nikto
```

```
docker run --rm -v $PWD:/output sullo/nikto   -host https://staging.tu-
dominio.com -Tuning 123bde -Display V -o /output/nikto_report.html -Format
html

# sqlmap (GET)
docker run --rm -v $PWD:/work sqlmapproject/sqlmap   -u "https://staging.tu-
dominio.com/item.php?id=1" --batch --risk=3 --level=5 --dbms=MariaDB --random-
agent

# sqlmap (request capturada con ZAP/Burp)
docker run --rm -v $PWD:/work sqlmapproject/sqlmap -r /work/login_request.txt
--batch --risk=3 --level=5 --smart
```

6.9 Seguridad – Análisis de Vulnerabilidades (OpenVAS / Nessus)

Escaneos autenticados de red y hosts.

```
# OpenVAS/GVM (ejemplo CLI; adaptar a su instalación)
gvm-cli ssh --gmp-username admin --gmp-password '***' --command "create_target
name=SwarmNodes hosts=10.0.0.0/24"
# Nessus (export de reportes)
nessuscli report --list
nessuscli report --export --format csv --output scans.csv --report-id <ID>
```

6.10 Seguridad – Verificación de Configuración Segura (Lynis, Docker Bench)

Auditar host y configuración de Docker.

```
# Lynis (host)
sudo lynis audit system --quick --report-file /tmp/lynis-report.dat

# Docker Bench for Security
docker run -it --rm --net host --pid host --userns host --cap-add
audit_control   -v /etc:/etc:ro -v /usr/bin/docker-containerd:/usr/bin/docker-
containerd:ro   -v /usr/bin/containerd:/usr/bin/containerd:ro -v
/var/lib:/var/lib:ro   -v /var/run/docker.sock:/var/run/docker.sock:ro -v
/etc/passwd:/etc/passwd:ro   docker/docker-bench-security | tee docker-
bench.log
```

6.11 Red y TLS (exposición, puertos, ciphers, headers)

Comprobaciones de superficie de ataque y endurecimiento de TLS/HTTP.

```
# Puertos abiertos
nmap -sV -p- staging.tu-dominio.com

# TLS y cadena de certificados
openssl s_client -connect staging.tu-dominio.com:443 -servername staging.tu-
dominio.com </dev/null | openssl x509 -noout -text
```

```
# Cabeceras de seguridad
curl -I https://staging.tu-dominio.com | egrep 'Strict-Transport-
Security|Content-Security-Policy|X-Frame-Options|X-Content-Type-
Options|Referrer-Policy'
```

6.12 Pruebas de Caos (netem, stress-ng)

Inyectar latencia/pérdida y carga de CPU/memoria. SOLO EN STAGING.

```
# Añadir latencia y pérdida de paquetes en interfaz (ejemplo eth0)
sudo tc qdisc add dev eth0 root netem delay 120ms 20ms loss 1%
# Quitar
sudo tc qdisc del dev eth0 root

# Carga de CPU/Memoria
sudo apt-get install -y stress-ng
stress-ng --cpu 4 --vm 2 --vm-bytes 70% --timeout 300s
```

6.13 Observabilidad (Logs, Métricas y Alertas)

Verificar integridad de logging y paneles (Grafana/Prometheus/ELK).

```
# Logs de servicios (Docker)
docker service logs -f web_app
# Métricas rápidas (cURL a /metrics si se expone)
curl -s http://prometheus:9090/metrics | head -n 20
```

7. Criterios de Aceptación por Área

- Funcional: 0 errores bloqueantes en flujos críticos; regresión OK.
- Rendimiento: P95 <= SLO; error rate < 1%.
- Alta Disponibilidad: RTO < 30s en failover; sin pérdida de sesiones.
- Seguridad: 0 críticas/altas; medias con plan ≤ 30 días.
- Datos: Sin corrupción; replicación en verde tras eventos.

8. Evidencia y Reporte

Guardar HTML/CSV/JTL/LOG por ejecución. Capturas de dashboards. Reporte final con descripción, impacto, CVSS, PoC (si aplica), recomendación, responsable y fecha objetivo.

9. Cronograma de Ejecución (Referencia)

Semana 1: Funcional + baseline rendimiento.
Semana 2: Pentesting (ZAP/Nikto/sqlmap) + hardening inicial.
Semana 3: Stress/Soak + failover + caos + FS y DB.
Semana 4: Vulnerabilidades (OpenVAS/Nessus) + re-pruebas + cierre.

10. Apéndices de Configuración

HAProxy – ejemplo de rate limiting con stick-table:

```
backend web
```

```
  stick-table type ip size 1m expire 10m store gpc0,conn_rate(10s)
  tcp-request content track-sc0 src
  acl abuse sc0_conn_rate gt 100
  http-request deny if abuse
```

NGINX – cabeceras de seguridad recomendadas:

```
add_header X-Frame-Options SAMEORIGIN always;
add_header X-Content-Type-Options nosniff always;
add_header Referrer-Policy no-referrer-when-downgrade always;
add_header Content-Security-Policy "default-src 'self' https: data:;" always;
add_header Strict-Transport-Security "max-age=63072000; includeSubDomains;
preload" always;
```